1、设置断点

在程序里面放置一个断点，也就是双击需要放置断点的程序左边的栏目上。

2、调试

(1)点击”打开透视图”按钮，选择调试透视图，则打开调试透视图界面，然后先设置断点，按调试按钮则可以调试程序。

(2)或者直接点击调试按钮，那个虫子样子的按钮，然后就会进入调试透视图的界面。前提你需要是要在程序中设置好断点。

(3)或者右键单击该程序，选择debug as（调试方式） -> java application（Java应用程序），则可以调试程序了。前提你需要是要在程序中设置好断点。

3、调试快捷键

以debug方式运行java程序后

（F5）单步执行程序，遇到方法时进入；

（F6）单步执行程序，遇到方法时跳过；

（F7）单步执行程序，从当前方法跳出；

（F8）直接执行程序。遇到断点时暂停；

按F5就是一层层深入的debug相当于剥葱样，会进入每个方法内部；

按F6就是在当前函数一步步debug，不理会深层次运作；

按F7就是如果当前进入了某个方法内部，都跳转到该方法的结尾代码处；

按F8就是如果后面还有断点，则运行到下一断点处，否则运行完程序；

另，在debug时，会有很多有用信息显示在debug框里，如堆栈信息，需要自己实践；在程序界面里，鼠标移到变量上时会有当前变量的属性值。

4、改变变量的值

在变量窗口中右击变量，在弹出的菜单中选择[change value]，弹出修改画面。

改变其值后，按下ok按钮；这时变量的值就改为修改后的值了。

5、检查代码段执行结果

在程序运行的时候，想看看某段程序运行的结果是否正确，或者查看某个变量的值时：

选择一句或一段代码，右键单击在弹出的菜单中选择Watch（创建查看表达式）项，此时，代码的执行结果将显示在Expressions（表达式）窗口中。

选择一句或一段代码，右键单击在弹出的菜单中选择Inspect（检查）项，可以直接显示表达式的值。

1.在你觉得有错的地方设置断点。

在代码行数前，点击右键，注意是右键，然后选择Toggle Breakpoint。

你可能会问，我如何知道在哪儿放置断点？

如果对这个问题完全没有感觉，你完全可以多打几个断点，单步调试直到找到异常，只是多花一点时间而已，而且这样可以更深入了解程序的执行过程！

当然，如果你大致可以直到哪里可能会出问题，或者异常信息报告类位置，那么就可以在这里设置断点。

2.点击Debug，如果是web程序，需要你将Tomcat或者Apache服务器以Debug模式启动。

这很重要，标准的Start模式，不能进入预先设置的断点，也就不能达到调试的目的。

3.运行程序，当程序运行到刚才设置断点的位置就会停下来，并且那行代码底色会高亮显示。

作用域 功能 快捷键

全局 单步返回 F7

全局 单步跳过 F6

全局 单步跳入 F5

全局 单步跳入选择 Ctrl+F5

全局 调试上次启动 F11

全局 继续 F8

全局 使用过滤器单步执行 Shift+F5

全局 添加/去除断点 Ctrl+Shift+B

全局 显示 Ctrl+D

全局 运行上次启动 Ctrl+F11

全局 运行至行 Ctrl+R

全局 执行 Ctrl+U

4.进入调试界面可以看到你想要的信息。

5.在Variables里面可以查看所有变量的值，比如刚才设置的断点里面的值，右键ChangeValue可以更改，部分IDE支持在窗口中热更改并执行代码。

补充：

Java断点快捷键

1、Ctrl+M –切换窗口的大小 2、Ctrl+Q –跳到最后一次的编辑处

3、F2 –当鼠标放在一个标记处出现Tooltip时候按F2则把鼠标移开时Tooltip还会显示即Show Tooltip Description。 F3 –跳到声明或定义的地方。 F5 –单步调试进入函数内部。

F6 --单步调试不进入函数内部,如果装了金山词霸2006则要把“取词开关”的快捷键改成其他的。

F7 --由函数内部返回到调用处。 F8 --一直执行到下一个断点。

1

2

3

4、Ctrl+Pg~ –对于XML文件是切换代码和图示窗口 5、Ctrl+Alt+I –看Java文件中变量的相关信息

6、Ctrl+PgUp –对于代码窗口是打开“Show List”下拉框,在此下拉框里显示有最近曾打开的文件

7、Ctrl+/ –在代码窗口中是这种//~注释。

Ctrl+Shift+/ –在代码窗口中是这种/~/注释,在JSP文件窗口中是〈!–~–〉。 8、Alt+Shift+O(或点击工具栏中的Toggle Mark Occurrences按钮) 当点击某个标记时可使本页面中其他地方的此标记黄色凸显,并且窗口的右边框会出现白色的方块,点击此方块会跳到此标记处。

9、右击窗口的左边框即加断点的地方选Show Line Numbers可以加行号。 10、Ctrl+I格式化激活的元素Format Active Elements。 Ctrl+Shift+F格式化文件Format Document。 11、Ctrl+S保存当前文件。

Ctrl+Shift+S保存所有未保存的文件。

12、Ctrl+Shift+M(先把光标放在需导入包的类名上) 作用是加Import语句。 Ctrl+Shift+O作用是缺少的Import语句被加入,多余的Import语句被删除。 13、Ctrl+Space提示键入内容即Content Assist,此时要将输入法中

Chinese(Simplified)IME-Ime/Nonlme Toggle的快捷键(用于切换英文和其他文字)改成其他的。

Ctrl+Shift+Space提示信息即Context Information。 14、双击窗口的左边框可以加断点。 15、Ctrl+D删除当前行。 其它:

Ctrl+1 快速修复(最经典的快捷键,就不用多说了)

Java数据结构快速入门Java数据结构基…Java数据结构基…

Ctrl+D: 删除当前行

Ctrl+Alt+↓ 复制当前行到下一行(复制增加) Ctrl+Alt+↑ 复制当前行到上一行(复制增加)

Alt+↓ 当前行和下面一行交互位置(特别实用,可以省去先剪切,再粘贴了) Alt+↑ 当前行和上面一行交互位置(同上) Alt+← 前一个编辑的页面

Alt+→ 下一个编辑的页面(当然是针对上面那条来说了) Alt+Enter 显示当前选择资源(工程,or 文件 or文件)的属性

Shift+Enter 在当前行的下一行插入空行(这时鼠标可以在当前行的任一位置,不一定是最后) Shift+Ctrl+Enter 在当前行插入空行(原理同上条) Ctrl+Q 定位到最后编辑的地方

Ctrl+L 定位在某行 (对于程序超过100的人就有福音了) Ctrl+M 最大化当前的Edit或View (再按则反之) Ctrl+/ 注释当前行,再按则取消注释 Ctrl+O 快速显示 OutLine Ctrl+T 快速显示当前类的继承结构 Ctrl+W 关闭当前Editer

Ctrl+K 参照选中的Word快速定位到下一个

Ctrl+E 快速显示当前Editer的下拉列表(如果当前页面没有显示的用黑体表示) Ctrl+/(小键盘) 折叠当前类中的所有代码 Ctrl+×(小键盘) 展开当前类中的所有代码

Ctrl+Space 代码助手完成一些代码的插入(但一般和输入法有冲突,可以修改输入法的热键,也可以暂用Alt+/来代替)

Ctrl+Shift+E 显示管理当前打开的所有的View的管理器(可以选择关闭,激活等操作) Ctrl+J 正向增量查找(按下Ctrl+J后,你所输入的每个字母编辑器都提供快速匹配定位到某个单词,如果没有,则在stutes line中显示没有找到了,查一个单词时,特别实用,这个功能Idea两年前就有了)

Ctrl+Shift+J 反向增量查找(和上条相同,只不过是从后往前查)

Ctrl+Shift+F4 关闭所有打开的Editer Ctrl+Shift+X 把当前选中的文本全部变味小写 Ctrl+Shift+Y 把当前选中的文本全部变为小写 Ctrl+Shift+F 格式化当前代码

Ctrl+Shift+P 定位到对于的匹配符(譬如{}) (从前面定位后面时,光标要在匹配符里面,后面到前面,则反之)

下面的快捷键是重构里面常用的,本人就自己喜欢且常用的整理一下(注:一般重构的快捷键都是Alt+Shift开头的了)

Alt+Shift+R 重命名 (是我自己最爱用的一个了,尤其是变量和类的Rename,比手工方法能节省很多劳动力)

Alt+Shift+M 抽取方法 (这是重构里面最常用的方法之一了,尤其是对一大堆泥团代码有用) Alt+Shift+C 修改函数结构(比较实用,有N个函数调用了这个方法,修改一次搞定) Alt+Shift+L 抽取本地变量( 可以直接把一些魔法数字和字符串抽取成一个变量,尤其是多处调用的时候)

Alt+Shift+F 把Class中的local变量变为field变量 (比较实用的功能) Alt+Shift+I 合并变量(可能这样说有点不妥Inline) Alt+Shift+V 移动函数和变量(不怎么常用) Alt+Shift+Z 重构的后悔药(Undo)

另一篇

作为开发人员，掌握开发环境下的调试技巧十分有必要。去年就想把关于Eclipse断点调试总结下了，由于对时间的掌控程度仍需极大提高，结果拖到今年才写了此篇博文难过。关于java调试技术还有很多，如Java Debug Interface等，依据具体项目的需要，还有很多值得去研究和学习的。该博文仅就Eclipse断点调试技巧做下总结，不足够的地方还请大牛们指点。

1 Debug视图

1.1 线程堆栈视图

线程堆栈视图表示当前线程的堆栈，从中可以看出在运行哪些代码，并且整个调用过程，以及代码行号。分别介绍一下这几个按钮的含义。从左至右分别为：

1

1.表示当前实现继续运行直到下一个断点，快捷键为F8。

2.表示打断整个进程

3.表示进入当前方法，快捷键为F5。

4.表示运行下一行代码，快捷键为F6。

5.表示退出当前方法，返回到调用层，快捷键为F7。

6、表示返回当前线程堆栈的起始处开始执行。

1.2 变量视图

1.为变量名视图，显示当前代码行中所有可以访问的实例变量和局部变量

2.显示所有的变量值

3.可以通过该窗口来改变变量值

1.3 断点视图

1.显示所有断点

将当前窗口1中选中的端口失效，再次点击启用。

3.异常断点

经常遇见一些异常，然后程序就退出来了，要找到异常发生的地方就比较难了，还好可以打一个异常断点。到了异常的地方，当前线程就挂起。

1.4 表达式视图

表达式视图是Debug过程中较为常用的一个视图，可以对自己的感兴趣的一些变量进行观察，也可以增加一些自己的表达式，也可以查看一行代码的运行结果。1.表达式2. 点击此可以新增一个表达式。

2 断点种类

2.1 方法断点

断点大家都比较熟悉，在Eclipse Java 编辑区的行头双击就会得到一个断点，代码会运行到此处时停止。方法断点是打在方法的入口处。

方法断点的特别之处在于它可以打在 JDK的源码里，由于 JDK 在编译时去掉了调试信息，所以普通断点是不能打到里面的，但是方法断点却可以，可以通过这种方法查看方法的调用栈。

2.2 条件断点

条件断点，顾名思义就是一个有一定条件的断点，只有满足了用户设置的条件，代码才会在运行到断点处时停止。

在断点处点击鼠标右键，选择最后一个”Breakpoint Properties”

断点属性中有三个选择，启用“Enable”,Hit Count和Conditional中依据我们的需要来选择。

1）Hit Count 是指定断点处的代码段运行多少次，最典型的就是循环，如果要让一个循环执行10次就线程挂起，则指定Hit Count值为10，那么当前的循环执行到第九次的时候就会挂掉。

2）Conditional，顾名思义，就是条件判断，例如我们需要循环变量i==10时，线程挂起，则条件设定为i==10,选择Suspend when “true”。

那如果上面的Hit Count和Conditional都选择的话，如果表达式和值设置不合理则会实效。如果选择Suspend when value changes，那么可能在Conditional在变量值发生改变的时候就挂起。

2.3 变量断点

断点不仅能打在语句上，变量也可以接受断点，

上图就是一个变量的打的断点，在变量的值初始化，或是变量值改变时可以停止，当然变量断点上也是可以加条件的，和上面的介绍的条件断点的设置是一样的。呵呵，这部分我没去实践，找不到在哪里设置~

2.4 修改变量值

修改变量值的情况适用于代码停在断点处，但是传过来的值不正确，那么可以通过修改这个变量值，保证代码走正确的流程；或者是说有一个异常分支老是进不去，能不能调试时改一下条件，看一下异常分支代码是否正确。

在Debug 视图的 Variables 小窗口中，我们可以在变量上右键，选择”Change Value…” 在弹出的对话框中修改变量的值。

或是在下面的值查看窗口中修改，保用Ctr+S 保存后，变量值就会变成修改后的新值了。

2.5 重新调试

这种调试的回退不是万能的，只能在当前线程的栈帧中回退，也就说最多只能退回到当前线程的调用的开始处。回退时，请在需要回退的线程方法上点右键，选择 “Drop to Frame”

————————————————

版权声明：本文为CSDN博主「zzzzzzzhu」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/zzzzzzzhu/java/article/details/82556335>

https://blog.csdn.net/zzzzzzzhu/article/details/82556335